

Quatrième partie

L'interblocage



Contenu de cette partie

- Notion d'interblocage
- Caractérisation des situations d'interblocage
- Protocoles de traitement de l'interblocage
 - préventifs
 - curatifs
- Apport déterminant d'une bonne modélisation/formalisation pour la recherche de solutions



Plan

- 1 Propriétés
 - Sûreté / vivacité
 - La vivacité
 - La famine
- 2 L'allocation de ressources multiples
- 3 L'interblocage
 - Graphe d'allocation
 - Prévention
 - Détection
 - Guérison
- 4 Conclusion



Propriétés temporelles

Pour *toutes* les exécutions possibles,
à *tout* moment d'une exécution.

- sûreté : rien de mauvais ne se produit
l'exclusion mutuelle, les invariants d'un programme
- vivacité : quelque chose de bon finit par se produire
l'équité, l'absence de famine, la terminaison d'une boucle
- (possibilité : pour *certaines* exécutions, ...)



La vivacité

- **Progression** : si *des* activités déposent des requêtes de manière continue, *une* des requêtes finira par être satisfaite ;
- **Vivacité faible** : si une activité dépose sa requête de manière continue, elle finira par être satisfaite ;
- **Vivacité forte** : si une activité dépose une requête infiniment souvent, elle finira par être satisfaite ;
- **Vivacité FIFO** : si une activité dépose une requête, elle sera satisfaite avant tout autre requête (conflictuelle) déposée ultérieurement.

Facile mais peu performante en centralisé, difficile en réparti.

(les activités sont supposées se comporter « correctement »)



La famine (privation)

Définition

Une activité est en famine lorsqu'elle attend infiniment longtemps la satisfaction de sa requête (elle n'est jamais satisfaite).

Vivacité \rightarrow absence de famine \rightarrow absence d'interblocage



Plan

- 1 Propriétés
 - Sûreté / vivacité
 - La vivacité
 - La famine
- 2 L'allocation de ressources multiples
- 3 L'interblocage
 - Graphe d'allocation
 - Prévention
 - Détection
 - Guérison
- 4 Conclusion



Allocation de ressources multiples

- Ressources banalisées, réutilisables, regroupées en classes
- Une activité demande un certain nombre de ressources dans chaque classe
- Demande **bloquante**, ressources allouées par un gérant de ressources
- Interface du gérant :
 - demander : $(\text{IdClasse} \rightarrow \text{natural}) \rightarrow (\text{Set of IdRessource})$
 - libérer : $(\text{Set of IdRessource}) \rightarrow \text{unit}$
- Le gérant :
 - rend la ressource réutilisable, lors d'une libération ;
 - libère les ressources détenues, à la terminaison d'une activité.



Exemples

Sémaphore

1 classe, R ressources, demande = 1

Philosophes

N classes de 1 ressource, P_i demande R_i et $R_{i\oplus 1}$

Allocateur mono-classe

1 classe de R ressources, demande $\in [1..R]$

Lecteurs/rédacteurs

1 classe de N ressources, demande = 1 ou = N

Plan

- 1 Propriétés
 - Sûreté / vivacité
 - La vivacité
 - La famine
- 2 L'allocation de ressources multiples
- 3 L'interblocage
 - Graphe d'allocation
 - Prévention
 - Détection
 - Guérison
- 4 Conclusion



L'interblocage

Allocation de ressources réutilisables

- non réquisitionnables
- non partageables
- en quantités entières et finies
- uniques ou multiples

Problème : A_1 demande R_1 puis demande R_2 ,
 A_2 demande R_2 puis demande R_1
entrelacement \rightarrow risque d'interblocage.

- 1 A_1 demande et obtient R_1
- 2 A_2 demande et obtient R_2
- 3 A_1 demande R_2 et se bloque
- 4 A_2 demande R_1 et se bloque

Définition de l'interblocage

Interblocage : définition

Un **ensemble** d'activités est en *interblocage* (*deadlock*) si et seulement si **toute** activité de l'ensemble est **en attente** d'une ressource qui ne peut être libérée que par une autre activité de l'ensemble.

L'interblocage est un état stable.



Graphe d'allocation

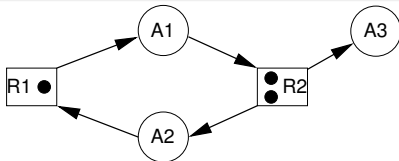
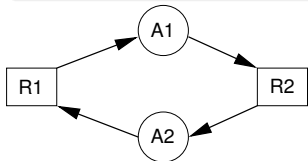


A est en attente de R



A possède R

Interblocage \equiv cycle/knot dans le graphe d'allocation



Conditions nécessaires à un interblocage

- 1 Les ressources sont utilisées en exclusion mutuelle
- 2 Les activités demandent plusieurs ressources en plusieurs fois :
 - pas de libération nécessaire avant une nouvelle demande
 - blocage tant que la requête ne peut pas être satisfaite
- 3 Ressources non réquisitionnables
- 4 Attente circulaire (cycle dans le graphe d'allocation)

Solutions

- Prévention : empêcher l'une des quatre conditions de survenir
- Détection + guérison : détecter l'interblocage, et l'éliminer



Prévention

Éviter l'accès exclusif

Ressource virtuelle : imprimante, fichier

Éviter la redemande bloquante

- Allocation globale : demander en une seule fois toutes les ressources nécessaires
 - connaissances a priori des ressources nécessaires
 - sur-allocation et risque de famine
- Acquisition non bloquante : le demandeur peut ajuster sa demande



Prévention

Éviter la non-réquisition

Avant de demander de nouvelles ressources, l'activité est tenue de libérer toutes les ressources dont elle dispose (quitte à les remettre dans la nouvelle demande).

(La libération de toutes les ressources n'est en réalité nécessaire que si la demande de l'activité est bloquante)

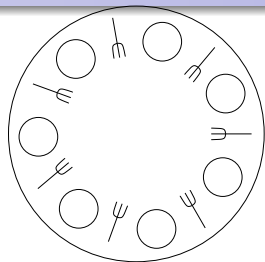
Éviter l'attente circulaire : classes ordonnées

- Un ordre est défini sur les classes de ressources
- Toute activité doit demander les ressources selon cet ordre

Solution utilisée pour éviter l'interblocage dû à des verrous multiples.

Exemple : Philosophes et spaghettis – Dijkstra

N philosophes sont autour d'une table.
Il y a une assiette par philosophe, et
une fourchette entre chaque assiette.
Pour manger, un philosophe doit
utiliser les deux fourchettes **adjacentes**
à son assiette (et celles-là seulement).



Un philosophe peut être :

- penseur : il n'utilise pas de fourchettes ;
- mangeur : il utilise les deux fourchettes adjacentes ; aucun de ses voisins ne peut manger ;
- demandeur : il souhaite manger mais ne dispose pas des deux fourchettes.

Ce problème est analogue au problème de l'allocateur multi-classes multi-ressources.



Exemple : philosophes et interblocage

Risque d'interblocage

Chaque philosophe demande sa fourchette gauche et l'obtient. Puis quand tous ont leur fourchette gauche, chaque philosophe demande sa fourchette droite et se bloque. \Rightarrow interblocage

Solutions

Allocation globale : chaque philosophe demande simultanément les deux fourchettes.

Non conservation : quand un philosophe essaye de prendre sa seconde fourchette et qu'elle est déjà prise, il relâche la première et se met en attente sur la seconde.

Classes ordonnées : imposer un ordre sur les fourchettes \equiv tous les philosophes prennent d'abord la gauche puis la droite, sauf un qui prend d'abord droite puis gauche.

Esquive

Avant toute allocation, évaluation du risque (futur) d'interblocage.

L'algorithme du banquier

- Chaque activité annonce le nombre maximal de ressources qu'elle possédera.
- L'algorithme maintient le système dans un état fiable, i.e. tel qu'il existe toujours une possibilité d'éviter l'interblocage dans le pire des cas.
- En cas de danger détecté, le requête est mise en attente (comme si les ressources n'étaient pas disponibles).
- Un état est fiable si pour toute activité A_i , les ressources que A_i pourra demander peuvent être satisfaites avec les ressources actuellement disponibles + les ressources détenues par les $A_j, j < i$.

12 ressources, $A_0/A_1/A_2$ annoncent 10/4/9

	max	poss.	dem	
A_0	10	5		
A_1	4	2	+1	oui $(5 + \mathbf{4} + 2 \leq 12) \wedge (\mathbf{10} + 2 \leq 12) \wedge (\mathbf{9} \leq 12)$
A_2	9	2	+1	non $(\mathbf{10} + 2 + 3 > 12)$ $\wedge (5 + 2 + \mathbf{9} > 12)$ $\wedge ((5 + \mathbf{4} + 3 \leq 12) \wedge (\mathbf{10} + 3 > 12) \wedge (5 + \mathbf{9} > 12))$

Détection

- Construire le graphe d'allocation
- Détecter l'existence d'un cycle (ressources uniques) ou d'un « knot » (ressources multiples équivalentes)
 - knot = composante fortement connexe terminale
 - = ensemble S d'activités/ressources tels que
 - $\forall s \in S : \text{successeur}(s) \in S$
 - $\wedge s \in \text{activité} \Rightarrow \text{successeur}(s) \neq \emptyset$
 - $\wedge s \in \text{ressources} \Rightarrow \text{card}(\text{succ}(s)) = \text{nb ress.}$
- Algorithme coûteux \rightarrow périodiquement (et non à chaque allocation)



Guérison

Réquisition des ressources détenues par une (ou plusieurs) activité(s).

- Comment choisir le(s) activités victimes (la dernière qui a alloué, la plus grosse/petite consommatrice, notion d'importance. . .) ?
- Annulation de l'activité ou retour en arrière ?
- Solution coûteuse (détection + choix + travail perdu), pas toujours acceptable (systèmes interactifs, systèmes embarqués).
- Simplifie l'allocation.
- Points de reprise pour retour arrière.



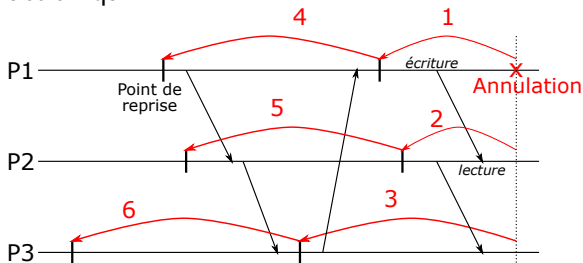
Points de reprise

Définition

Sauvegarde d'états intermédiaires pour éviter de perdre tout le travail.

Utilisé pour les transactions en base de données

Effet domino : l'annulation d'une action entraîne l'annulation d'une deuxième action qui...



Plan

- 1 Propriétés
 - Sûreté / vivacité
 - La vivacité
 - La famine
- 2 L'allocation de ressources multiples
- 3 L'interblocage
 - Graphe d'allocation
 - Prévention
 - Détection
 - Guérison
- 4 Conclusion



L'interblocage

- Usuellement : interblocage = inconvénient occasionnel
 - → laissé à la charge de l'utilisateur/du programmeur
 - utilisation de méthodes de prévention simples (p.e. classes ordonnées)
 - ou détection empirique et guérison par choix manuel des victimes
- Cas particuliers : systèmes ouverts (plus ou moins) contraints par le temps
 - systèmes interactifs, multiprocesseurs, systèmes embarqués
 - recherche de méthodes efficaces, prédictibles, ou automatiques
 - compromis/choix à réaliser entre la prévention (statique, coûteuse, restreint le parallélisme) et détection/guérison (moins prédictible, coûteuse quand les conflits sont fréquents).
- Émergence d'approches sans blocage (cf synchronisation non bloquante)



Autres difficultés de synchronisation

- Accès non protégé à un état partagé (conflits d'écriture, visibilité d'états transitoires). Se manifestent souvent par des comportements non reproductibles (heisenbugs)
Variables partagées \Rightarrow verrous
Outils d'analyse statique pour identifier de potentiels problèmes
- Invalidation d'un invariant (souvent facile à détecter, mais coûteux et non correctible en général)
- Famine : difficilement prouvable, impossible à détecter (sauf stratégie régulière : FIFO)
- « Livelock » : activité bouclant sans jamais libérer toutes ses ressources ou acquisition non bloquante avec tentatives qui échouent toujours.

